Università degli Studi di Salerno
Corso di Ingegneria del Software

CarFix
System Design Document
Versione 1.4

Data: 25/11/2024

Coordinatore del progetto:

Nome	Matricola

Partecipanti:

Nome	Matricola
Felice Chirico	0512108085
Carmine Nunziata	0512116290

Scritto da:	

Revision History

Data	Versione	Descrizione	Autore
21/11/2024	1.0	Prima stesura del documento	Chirico Felice
22/11/2024	1.1	aggiunta dei diagrami UML relativi all'architettura e al deployement di sistema	Carmine Nunziata
23/11/2024	1.2	aggiunta della tabella delle operazioni degli utenti e revisione obiettivi di progettazione	Entrambi i partecipanti
24/11/2024	1.3	revisione paragrafo sulle condizioni limite del sistema	Entrambi i partecipanti
25/11/2024	1.4	Impaginazione e revisione finale del documento	Entrambi i partecipanti

Indice

1. Introduzione
1.1 scopo del sistema

La piattaforma **CarFix** nasce dall'esigenza di semplificare e ampliare la vendita di pezzi di ricambio per veicoli, offrendo un servizio accessibile a un numero maggiore di clienti attraverso un sistema di e-commerce intuitivo e funzionale. Il sistema permette agli utenti di acquistare online componenti di qualità per la riparazione e manutenzione dei veicoli, con un'interfaccia semplice e accessibile da qualsiasi dispositivo dotato di un web browser.

CarFix si propone di fornire una piattaforma user-friendly che consenta agli utenti di consultare un vasto catalogo di prodotti, effettuare ricerche mirate per marca e modello, e completare l'acquisto in pochi click. I dati relativi a prodotti e ordini saranno gestiti tramite un database relazionale, garantendo un'esperienza d'uso rapida e affidabile.

La piattaforma include inoltre un sistema di autenticazione sicuro, basato su username e password, per proteggere i dati sensibili degli utenti e garantire che solo i clienti registrati possano accedere alle funzionalità avanzate del sistema, come la gestione degli ordini e lo storico degli acquisti. In questo modo, **CarFix** diventa uno strumento indispensabile per chi cerca praticità, sicurezza e qualità nel settore dei pezzi di ricambio per veicoli.

1.2 Obiettivi di progettazione

La piattaforma **CarFix** è progettata per offrire agli utenti un'esperienza di acquisto online di pezzi di ricambio per veicoli che sia rapida, affidabile e intuitiva. Gli obiettivi principali del sistema includono la creazione di un catalogo facilmente navigabile, la garanzia di prestazioni elevate anche durante picchi di traffico e la protezione dei dati sensibili degli utenti

1.2.1 Criteri di prestazione

Throughput (bassa priorità)
Il sistema deve essere in grado di supportare e gestire almeno cento utenti simultaneamente; è necessario quindi che il sistema sia scalabile

Tempo di risposta del sistema (media priorità)
Il tempo di caricamento delle pagine non deve superare i cinque secondi in condizioni di rete standard. Inoltre le operazioni di ricerca nel catalogo devono restituire i risultati entro 1 secondo per query.

Memoria (alta priorità)
Tutti i dati relativi al catalogo, ai magazzini, agli ordini effettuati e agli utenti registrati sono conservati in un database relazionale gestito, quale MySQL.

1.2.2 Criteri di affidabilità

Robustezza del sistema (alta priorità)
Robustness: il sistema deve essere in grado di gestire eventuali errori di immissione da parte dell'utente preservando l'integrità dei dati invitandolo a eseguire nuovamente l'operazione

Sicurezza (alta priorità)
il sistema deve garantire l'accesso alla modifica del catalogo esclusivamente agli utenti autorizzati; le operazioni di modifica sul catalogo non devono essere accedute in alcun modo da utenti che non abbiano i permessi da gestore

Aggiornamento dei dati (alta priorità)
Il sistema deve garantire che i dati salvati siano aggiornati dopo ogni transazione (es. dopo un ordine effettuato con successo, la disponibilità dei prodotti in magazzino deve essere aggiornata)

1.2.3 Criteri di supportabilità

Architettura Three-Tier (alta priorità)
Il sistema deve avere un'architettura a tre livelli che favorisca la una semplice manutenzione. L'idea è di usare un'architettura Three-Tier che organizza le applicazioni in tre tier di calcolo logici e fisici: il livello di presentazione, o interfaccia utente, il livello applicazione, dove i dati vengono elaborati, e il livello dati, dove i dati associati all'applicazione vengono memorizzati e gestiti.

Estensibilità (alta priorità)
Il sistema deve essere facilmente estensibile per permettere l'aggiunta di funzionalità in futuro; per garantire ciò il codice del sistema deve essere propriamente documentare per facilitare la manutenzione anche a sviluppatori esterni

1.2.4 Criteri di usabilità

Responsive (media priorità)
Il layout e i contenuti del sito devono adattarsi automaticamente alle dimensioni dello schermo dell'utente, garantendo una visualizzazione chiara e un'interazione agevole su dispositivi desktop, tablet e smartphone.

User friendly (media priorità)
L'interfaccia utente deve essere intuitiva e progettata per ridurre al minimo il numero di passaggi necessari per completare le operazioni principali, come la ricerca e l'acquisto di prodotti. Tutti gli elementi interattivi, come pulsanti e filtri, devono essere chiaramente identificabili e accompagnati da etichette o icone descrittive per agevolare la comprensione anche da parte di utenti meno esperti.

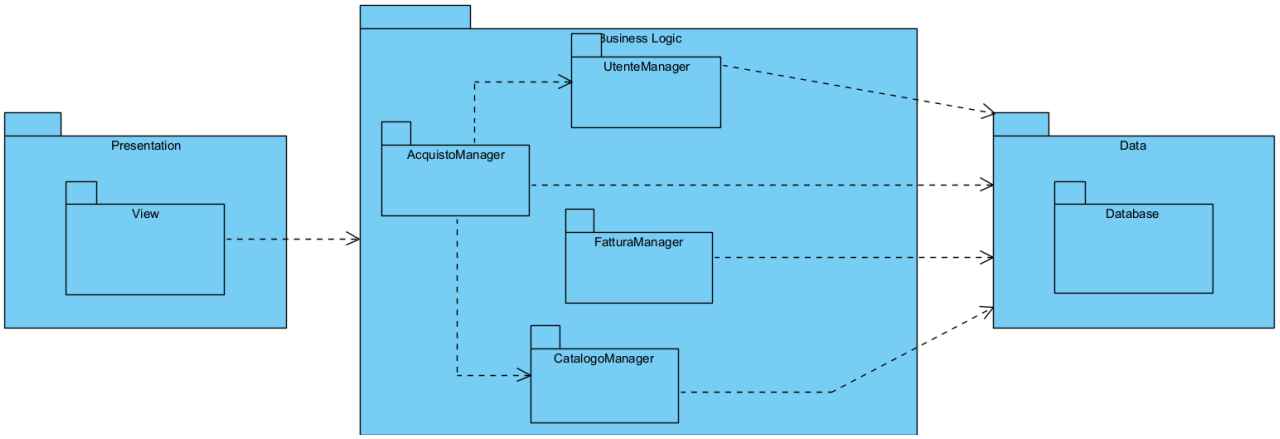
1.3 Riferimenti

R.A.D.: si farà riferimento ai requisiti non funzionali presenti nel R.A.D.

2. Architettura del sistema proposto

2.1 Decomposizione in sottosistemi

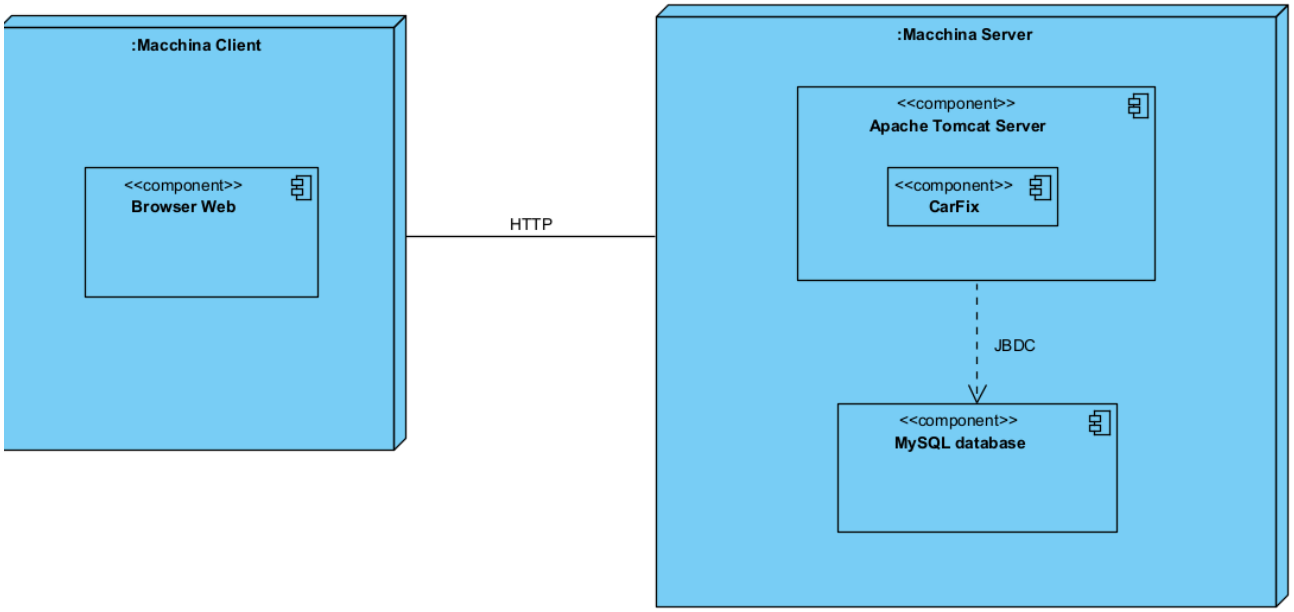
Il sistema è diviso in tre livelli: Presentation, Business logic e Data Storage.
il seguente diagramma UML mostra la suddivisione dei sottosistemi nei tre livelli e le loro relazioni



View: è il sottosistema che si occupa di gestire l'interfaccia utente.
AcquistoManagement: è il sottosistema che si occupa di gestire la logica dietro l'acquisto di uno o più prodotti: pagamento, carrello, aggiornamento del catalogo..
UtenteManagement: è il sottosistema che si occupa di gestire la logica dietro l'autenticazione dell'utente e i suoi permessi
CatalogoManagement: è il sottosistema che si occupa di gestire il catalogo del sistema.
Storage: è il sottosistema che si occupa dell'interazione con la base di dati.
FatturaManager: è il sottosistema che si occupa di controllare che le fatture emesse dal sito siano compatibili con i dati immessi dall'utente

2.2 Mapping Hardware/Software

L'architettura del sistema è Client-Server con la gestione dei dati persistenti delegata ad un'istanza di un DBMS relazionale, quale MySQL, che si trova sulla stessa macchina del server. La comunicazione tra il Browser, sulla Macchina Client, e il sistema, sulla Macchina Host, avviene attraverso richieste e risposte HTTP. Il deployment del sistema viene fatto su di un server Apache Tomcat installato sulla macchina Host, che comunica con il DBMS attraverso un driver JDBC.



2.3. Gestione della persistenza

i dati del sistema che devono essere resi persistenti sono:

- informazioni utente
- informazioni sui ricambi
- informazioni sul carrello
- informazioni sugli ordini e le fatture relative

il compito di salvare questi dati sarà affidato ad un istanza di MySQL.

2.4. Controllo degli accessi e sicurezza

Per prevenire accessi non autorizzati ad oggetti modificabili appartenenti all'entità del dominio, il sistema garantisce un controllo degli accessi basato su autenticazione tramite l'immissione di e-mail e password.

La tabella sottostante riporta le azioni che gli attori possono eseguire sugli oggetti dell'entità del dominio (nella lettura della tabella sottostante, si tenga presente che le funzionalità di "UtenteNonRegistrato" sono messe a disposizione di tutte le altre tipologie di utenti, quelle di "UtenteRegistrato" sono a disposizione anche di "GestoreMagazzino", "Gestorefatturazione".

Inoltre, si consideri il termine “Gestione” presente nella definizione di alcune operazioni della tabella, come rappresentativo di tre operazioni specifiche: “aggiunta”,”modifica”,”rimozione”.

Attori Oggetti	Utente non registrato	Utente registrato	Gestore Magazzino	Gestore Fatturazione
Utente	registrazione()	login() logout() visualizzaInf() modificaInf()	login() logout()	login() logout()
Carrello		aggiungiAlCarrello() rimuoviDalCarrello() visualizzaCarrello() modificaQtProd() acquista()		
Ordine		visualizzaOrdini()		visualizzaOrdini() filtraOrdini() modificaordine()
Prodotto	visualizzaProd()	visualizzaProd() recensisciProd()	visualizzaProd() modificaProd()	
Catalogo	visualizzaProd() cercaProd() filtraProd()	visualizzaProd() cercaProd() filtraProd()	aggiungiProd() rimuoviProd()	

2.5. Controllo globale del software

Il controllo globale del software è di tipo event-based. Essendo una webapp, sarà il Web Server ad occuparsi dello smistamento delle varie richieste HTTP verso delle apposite Servlet che si occuperanno di gestire la richiesta, interagire con le altre componenti del sistema ed elaborare una risposta. A fronte dell’elevato numero di utenti che potrebbe utilizzare la piattaforma in contemporanea e dell’elevata interattività della stessa si è optato per l’utilizzo del web server Tomcat, che supporta l’esecuzione di applicativi web basati su Java Servlets e pagine JSP e gestisce autonomamente ciascuna richiesta da un client creando un thread dedicato

2.6 Condizioni Limite

Installazione del sistema
L'installazione del sistema verrà eseguita da un programmatore che avrà anche il compito di riempire la base di dati con gli account dei vari gestori ed il catalogo.

Avvio del sistema
Il sistema verrà avviato da un sistemista in seguito all'installazione. Dopo l'avvio viene attivato il DBMS MySQL, la cui connessione col sistema verrà stabilita mediante driver JDBC.

Terminazione
Al momento della corretta chiusura dell'applicazione, si ha la terminazione del sistema con un regolare Log-out. Per consentire la corretta terminazione del server, l'amministratore del sistema dovrà effettuare la procedura di terminazione, dopo la quale nessun client potrà connettersi al sistema.